The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 9 Issue 1, 2021
pp: 12 - 18

# Obstacle Avoidance Robot

**[1]Ch Rajeshwari, [2]C. Uma Priyanka, [3]E. Hymavathi**
**[4]C. Navaneetha**
[1,2,3,4] UG Student, Department of ECE,
Dr K V Subba Reddy College Of Engineering For Women, Kurnool, Andhra Pradesh, India

**Abstract**
The goal of this project is to use a variety of autonomous navigation algorithms to avoid obstacles, allowing a robot to move and work in an unstructured, unknown environment. The investigation and study of the platform at the Mechatronics Laboratory, on which the navigation algorithm is implemented, is the first step. ROS has been utilized in relation to the software platform. An open-source framework for controlling the actions, tasks, and operations of robots is known as the Robot Operating System. Because the ROS-compatible TurtleBot3 (Burger) has been utilized. The evaluation of the various algorithms that are appropriate and pertinent to our objective, environment, and purpose is the second step. The navigation, which is typically divided into global motion planning and local motion control, can be implemented using a variety of methods. When autonomous mobile robots work in an environment, previous maps frequently contain inaccuracies or are incomplete. They require a safe course that avoids collision. As a result, the robot is able to move toward the open area while avoiding the obstacles thanks to the sensor that is mounted on it. This paper discusses three distinct Obstacle Avoidance algorithms for fully autonomous navigation in an unstructured environment. All of the algorithms are tested on the TurtleBot3 robot, where only LiDAR was used as a sensor to identify obstacles. The complexity of the algorithms grows as the evolution and the various possible situations in which the robot will have to move are considered. The main advantage of the third algorithm, "Autonomous Navigation," is that it can perform curved trajectories with precise path selection. Combining angular and linear velocity (980 different motions), the LiDAR scans 180 degrees in front of the robot to determine the correct direction. The automatic creation of the map is the final step. The official ROS environment software, RViz, will be used to analyze and compare this map to the one created using RViz. The tool can be used to record sensor data, debug problematic behaviors, visualize the robot's state and the performance of the algorithms, and more. This reactive approach to obstacle avoidance can be improved by driving robots successfully into challenging locations. In order to conclude this study, we will present the experimental results on TurtleBot3 to support the findings and make a case for the benefits and drawbacks

## 1. Introduction

. In the past, investigation into the development of unmanned air, underwater and land vehicles has been fundamentally the domain of military related organizations. Nowadays, the technological context, availability of precise sensors, the spread of open- source software and the increasing of computation power, has led the largest companies to take an interest on the

concept of automation and robotization and as a result autonomous navigation has become also one of hottest topics in the research‟s field.

In this thesis, we study the problem of autonomous navigation through an environment that is initially unknown, with the objective of reaching the farthest point in which the robot can move avoiding the obstacles. Without prior knowledge of the map, a moving robot must recognize its surroundings through onboard sensors and make instantaneous decisions to react to obstacles as they come into view. This problem lies at the intersection of several areas of robotics, including motion planning, perception, and exploration.

Different techniques could be used to implement the navigation that is generally separated into global motion planning and local motion control. The algorithms introduced in this work are linked to the local motion planning; therefore, using the sensor mounted on it, the robot is capable of avoiding the obstacles by moving toward the free area.

This document explains three possible algorithm solutions, based on Obstacle Avoidance, that address a complete autonomous navigation in an unstructured indoor environment.

The algorithms raise in complexity taking into consideration the evolution and the possible changed in which the robot will have to move, and all are tested on the TurtleBot3 robot (Waffle and Burger), where only LiDAR was used as sensor.

The implemented techniques necessitate the robot to select actions based on the construction of the environment that it has perceived. As we will observe in this thesis, standard motion planning techniques often limit performance to be conservative when deployed in unknown environments, where every unexplored region of the map may, in the worst case, poses hazard

To guarantee that the robot will not collide with potential obstacles, motion planners limit the robot's speed such that it could come to a stop, if need be, before the collisions.

The trajectory and the speed of the robot depend on many factors such as the type of floor, the limitations of the hardware, the size and the material of the wheels and the type of algorithm that manages the movement of the robot.

The map is built with two-dimensional Cartesian histogram grid based on the RViz software that is the official software used in ROS environment, which is updated continuously with range data sampled by onboard sensors.

In order to make this work more complete a different solution to the automatic creation of the map, has been proposed; this map will be analysed and compared with the one created by the ROS tool.

The result and some real applications of the algorithms are drawn in Chapter 6. Moreover, this chapter outline also the advantages/disadvantages and limitation of the algorithms. Finally, it proposes future approach and application as agricultural outdoor environment

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 9 Issue 1, 2021
pp: 12 - 18

**Fig.1** Morgan Quigley programmed the first iteration

## 2. Literature Review

ROS is an open-source, meta-operating system for the robot. It delivers the services you would imagine from an operating system, including hardware abstraction, low- level device control, implementation of commonly-used functionality, message- passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers, which has the target of simplifying the task of creating complex and robust robot behavior across a wide variety of robotic platforms.

Contrasting conventional operating systems, it can be used for several combinations of hardware implementation. Furthermore, it is considered as a robot software platform that offers various development environments specialized for developing robot application programs.

For example, consider a simple "retrieve an object" activity, in which a robot is required to retrieve a specific object. First of all, the robot must understand the request that means how to reach the goal. The robot must plan a sequence of actions to coordinate the object's search, which will require navigation through various rooms in a building, where the robot must be able to avoid all obstacles, optimizing the chosen path.

Once in a room, the robot must look for objects of similar size and find the required one. The robot must then return to its own steps and deliver the object to the desired position. Each of these sub problems can have an arbitrary number of issues; in the real world there are a lot of circumstance in each field that is difficult to predict and model, so no single individual can think to build a complete system from scratch.

So, ROS was built from the ground up to encourage collaborative robotics software development. In this example, a group might have specialists in indoor mapping and could contribute to a complex system for producing indoor maps; the same work could be done for an outdoor space (field or rows).

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 9 Issue 1, 2021
pp: 12 - 18

Another group may have experience in using maps to robustly navigate indoors, specialized in motion planning and Obstacle Avoidance. Another one may have discovered an approach to the vision that works with sensors able to offer capabilities such as gesture recognition, object recognition and scene recognition based on 3D depth information. ROS includes many features specifically designed to simplify this type of large-scale collaboration

The main features of ROS can be grouped in five characteristics:First is the reusability of the program. A user can focus on the goal related to its application that it would like to develop while downloading the corresponding package for the remaining functions. At the same time, he can share the program that he developed so that others can reuse it.

The second characteristic is that ROS is a communication-based program. Often, to provide a service, programs such as hardware drivers for sensors and actuators and features such as sensing, recognition and operating are developed in a single frame. However, to achieve the reusability of robot software, each program and feature is divided into smaller pieces based on its function. This is called componentization or modularization according to the platform.

The third is the support of development tools. ROS provides debugging tools, 2D visualization tool (such as Rqt) and 3D visualization tool (RViz) that can be used without developing the necessary tools for robot development. Tools that make it easy to visualize

Behaviours, and to record sensor data. A large and increasing gathering of robotics algorithms that allow you to map the environment, navigate around it, represent and interpret sensor data, plan motions, manipulate objects, and other operations is available.

For example, there are many occasions where a robot model needs to be visualized while developing a robot. The growing number of tools and their capabilities allows users not only to check the robot‟s model directly but also to perform a simulation using the provided 3D simulator (Gazebo).

The tool can also receive 3D distance information from cameras, as Intel Real Sense or Microsoft Kinect, and easily convert them into the form of point cloud, finally display them on the visualization tool.

The fourth is the active community. Ros is a community for an open source software platform. There are over 5,000 packages that have been voluntarily developed and shared as of 2017, the Wiki pages that document many of the aspects of the framework, and a question-and-answer site where you can ask for help and share what you‟ve learned.

The fifth is the construction of an ecosystem. Various software platforms have been developed and the most respected and used platform among them, ROS (for all the features that we already saw), is now shaping its ecosystem. It is creating an ecosystem for everyone: hardware developers from the robotic field such as a robot and sensor companies, ROS development operational team, application software developers, and users as the students, can be happy with it.

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 9 Issue 1, 2021
pp: 12 - 18

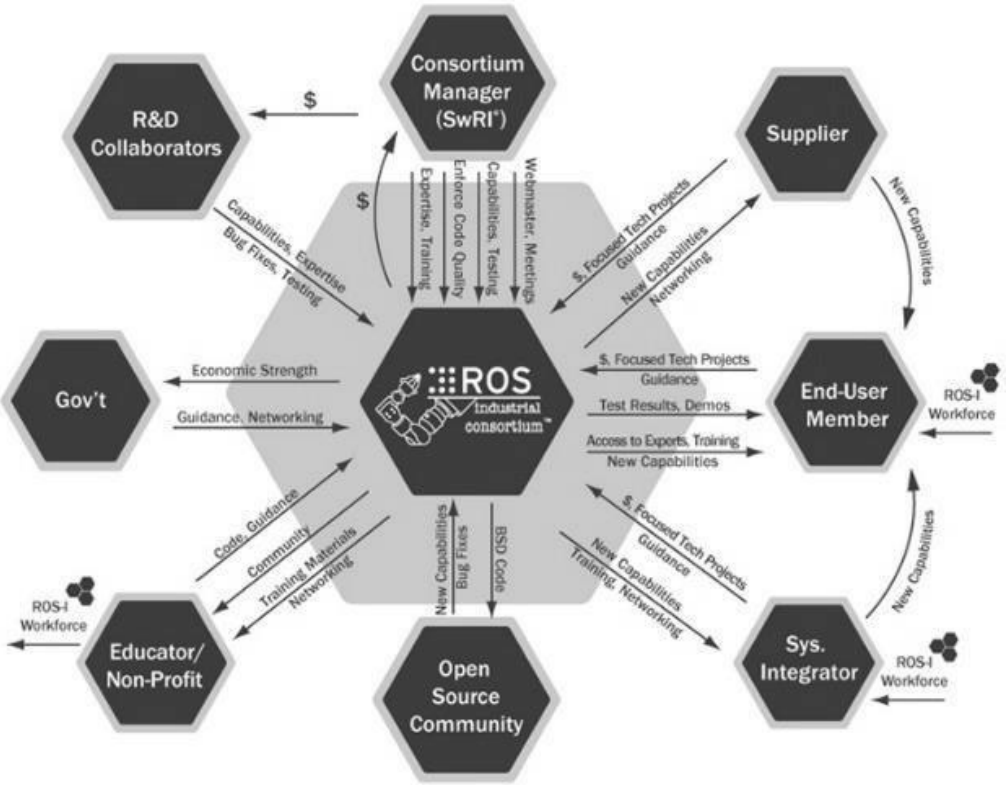


**Fig.2** Structure of ROS

## 3. Proposed System

A robot's sensors play an important role. There are a variety of methods and sensors that can be used to extract meaningful information from a variety of environments, use this information to identify the objects in the vicinity, and transmit it to the robot. Every piece of data that the robot can collect is used as data to carry out a task, develop a strategy, or provide input for some operation.

While there are many different kinds of sensors that can be used to gather this data, the distance sensor is the one that is utilized the most due to its simplicity and effectiveness. As distance sensors, laser-based ones like the LDS (Laser Distance Sensor), LiDAR (Light Detection and Ranging), or LRF (Laser Range Finders) and infrared-based ones like the Real Sense, Kinect, and Xtion are frequently utilized. In addition, there are a variety of sensors depending on the information that needs to be gathered, such as color cameras for object recognition, inertial sensors for estimating position, microphones for voice recognition, and torque sensors for controlling torque.

Each microprocessor is limited in the amount of data it can receive at any given time, but this varies depending on the type of sensor and the purpose for which it is being used. 1D and 2D sensors, as laser-based distance sensors do not transmit a lot of data, are more difficult for a microprocessor to handle than cameras, which transmit a lot of data and require a lot of processing power.

On ROS, there are a number of different sensor packages to choose from. Laser Distance Sensors (LDS) include a variety of sensors, including Light Detection and Ranging (LiDAR),

The Ciência & Engenharia - Science & Engineering Journal
ISSN: 0103-944X
Volume 9 Issue 1, 2021
pp: 12 - 18

Laser Range Finder (LRF), and Laser Scanner, as well as Pose Estimation (GPS + IMU), Cameras (which are commonly used for object and gesture recognition, face recognition, and 3D SLAM), Audio/Speech Recognition, and many others. 1D rangefinders are used for low-cost robots and include Infrared distance The LDS sensor uses a laser as its source to determine the distance to an object. Since the LDS guarantees high performance, high speed, and real-time data acquisition, it is frequently used in all systems that require distance measurement and a wide range of robotics fields. It is one of the most important sensors used in robotics to recognize people and objects from a distance.

If an obstacle is found, the LDS calculates the wavelength difference when the laser source is reflected by it. Control issues that can't be fixed by another sensor are the biggest issues that could arise with this kind of sensor because, for the low price, only one LDS is typically mounted on one device. A single laser source, a motor, and a reflective mirror make up a typical LDS. While scanning with the laser, the inner mirror is rotated by the motor. The LDS has a range of 180° to 360°.

The manner in which the laser is reflected on the environment by the tilted mirror is depicted in the first image from the left of Figure 3.2. While the motor rotates the mirror and scans the entire environment in the second and third images, the sensor records the returned laser and saves the return time (calculates the wavelength difference). Since the LDS sensor scans objects horizontally, closer objects are easier to identify, so accuracy decreases with increasing distance longer

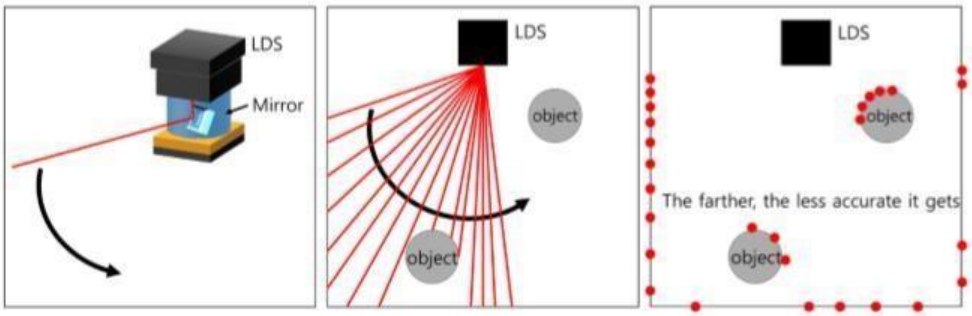


**Fig.3** Distance measurement using LDS

As we already say there are three official TurtleBot3 models Burger, Waffle and Waffle Pi. The basic components of TurtleBot3 are actuators, an SBC for operating ROS, a sensor or more than one for SLAM and navigation, restructure mechanism, an OpenCR embedded board used as the main controller, sprocket wheels that can be used with tire and caterpillar, and three cell lithium-poly battery.

TurtleBot3 Waffle is different from Burger in terms of a platform shape, which being bigger can conveniently mount many components and sensors, use of higher torque actuators that guarantee a maximum linear speed of 0.26 m/s and an angular velocityof 1.8 rad/s, high-performance SBC with Intel processor in terms of calculations, RealSense Depth Camera for object recognition and 3D SLAM.

Due to its characteristics, Waffle was the most used Robot during laboratory tests, also because from manual, it should have offered the possibility to use the Intel® Real SenseTM R200

camera. The Intel® Real SenseTM R200 camera should have provided depth and infrared video streams and the possibility to apply it for various applications such as gesture recognition, object recognition and scene recognition based on 3D depth information

## 4. Conclusion

In the beam of light, at the bottom left of the image, it is not a mistake, but it is due to the presence of a slit in the obstacles that bounded the perimeter, so the rays of the Lidar, mapping this area, shone through the small hole in the wall. The other image of Figure 6.3 is instead generated at the end of the simulation in the vineyard, using a script in python, in which the entire path executed by the robot, is represented from the starting point to the final one. It is possible to point out the presence of small errors of odometry, which are more evident when the robot performs curved trajectory and gradually spreads, decreasing the accuracy of the blue trace. This experiment was done in collaboration with another master student, who worked on a parallel project, consisting of identifying, given a generic map of the agricultural environment in the form of a binary matrix, the parcels within it in order to generate a path plan for the robot able to cover the entire environment with an optimal criterion

## References

1. YoonSeok Pyo, HanCheol Cho, RyuWoon Jung, TaeHoon Lim, ROS Robot Prog. Morgan Quigley, Brian Gerkey, and William D. Smart, Programming Robots with ROS, December 2015: First Edition, Published by O‟Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.
2. Masoud Nosrati * Ronak Karimi Hojat Allah Hasanvand, Investigation of the * (Star) Search Algorithms: Characteristics, Methods and Approaches, Vol (2), April 2012.
3. Nebot E., Bailey T., Guivant J., Navigation Algorithms for Autonomous Machines in Off-Road Applications, The University of Sydney, NSW 2006
4. A. Oualid Djekoune, Karim Achour and Redouane Toumi, A Sensor Based Navigation Algorithm for a Mobile Robot using the DVFF Approach, International Journal of Advanced Robotic Systems, Vol. 6, No. 2 (2009)
5. Bruno Siciliano, Oussama Khatib, Eds., Springer Handbook of robotics,
6. Springer-Verlag Berlin Heidelberg2016
7. Chuang Ruan, Jianping Luo, Yu Wu, Map navigation system based on optimal dijkstra algorithm, Proceedings ojCCIS2014
8. Javier Minguez, Associate Member, IEEE, and Luis Montano, Member, IEEE, Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios, ieee transactions on robotics and automation, vol. 20, no. 1, february 2004
9. Takahashi O, Schilling RJ (1989), Motion Planning in a Plane Using Generalized Voronoi Diagrams. IEEE Robotics and Automation.
10. Bhattacharya P, Gavrilova ML (2008), Roadmap-Based Path Planning-Using the Voronoi Diagram for a Clearance-Based Shortest Path. IEEE Robotics and Automation.